Foundations of Modern Networking

SDN, NFV, QoE, IoT, and Cloud

By: William Stallings

Chapter 4

SDN Data Plane and OpenFlow

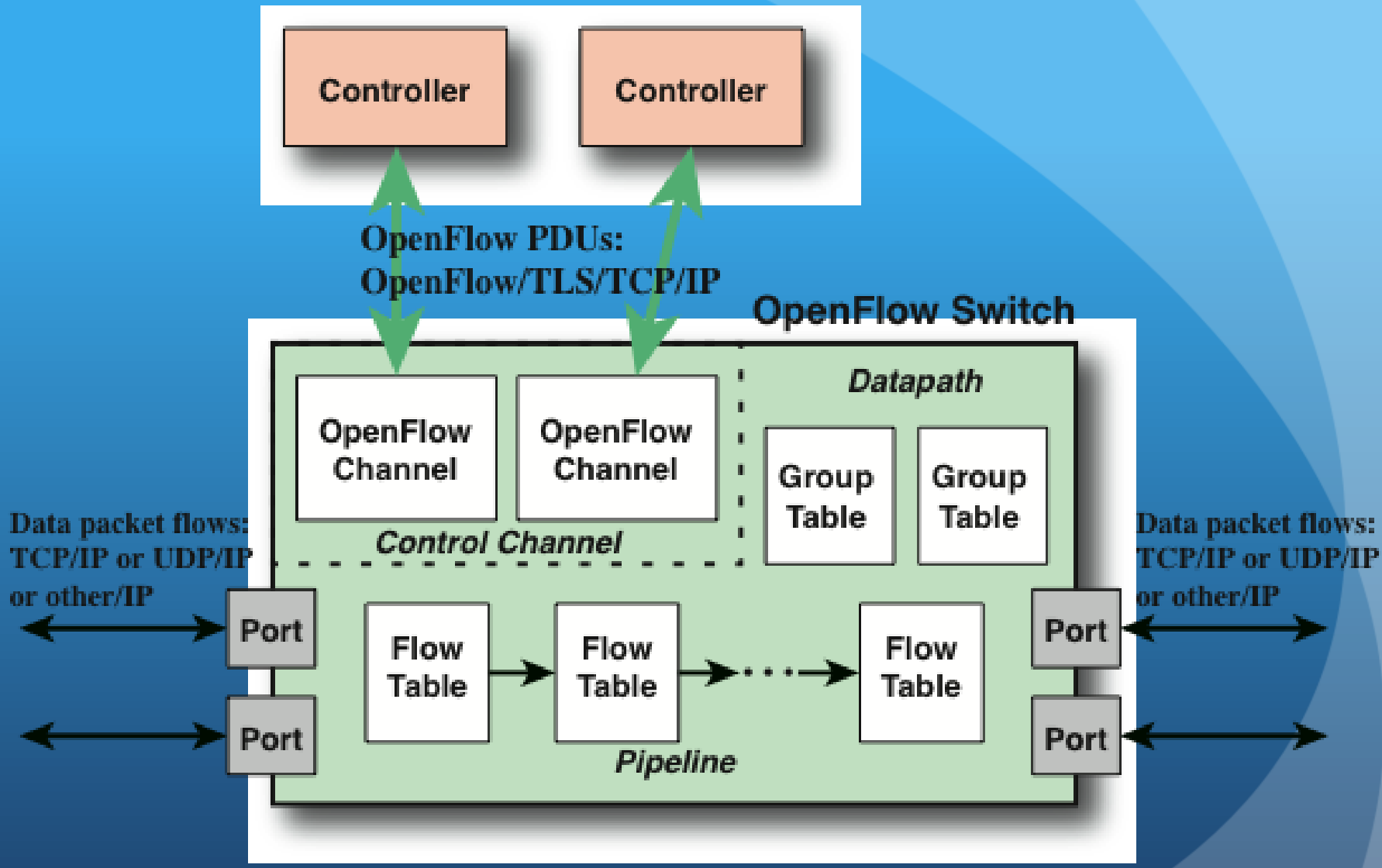


Figure 4.4 Main Components of an OpenFlow Switch

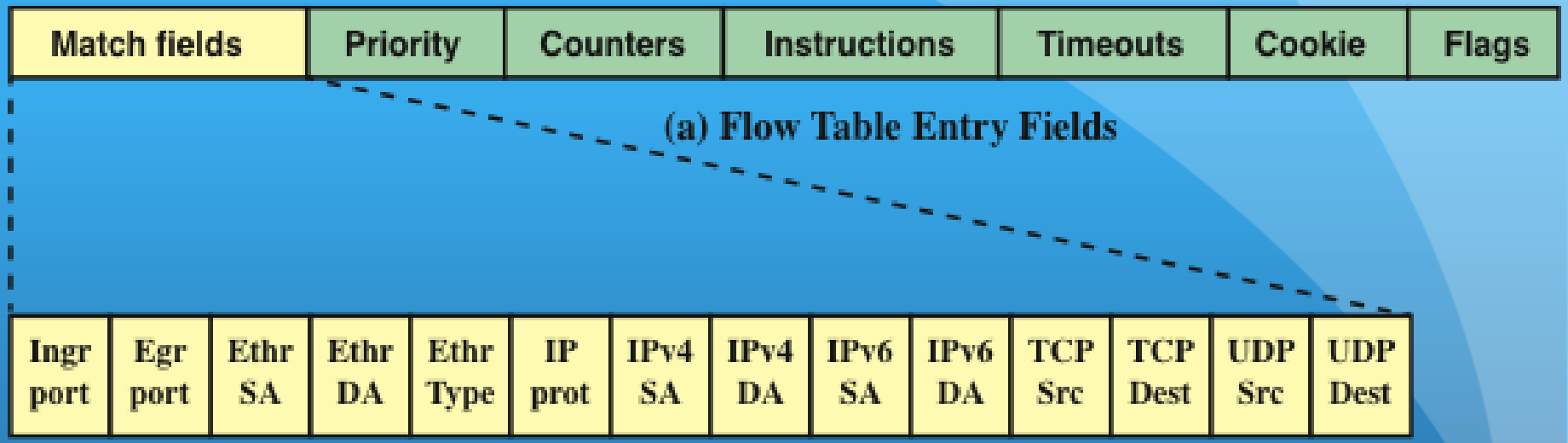


Figure 4.5 OpenFlow Table Entry Formats

Table 4.1 Required OpenFlow Counters

Counter	Usage	Bit Length
Reference count (active entries)	Per flow table	32
Duration (seconds)	Per flow entry	32
Received packets	Per port	64
Transmitted packets	Per port	64
Duration (seconds)	Per port	32
Transmit packets	Per queue	64
Duration (seconds)	Per queue	32
Duration (seconds)	Per group	32
Duration (seconds)	Per meter	32

Flow Table Pipeline

- A switch includes one or more flow tables
- If there is more than one flow table, they are organized as a pipeline, with the tables labeled with increasing numbers starting with zero
- The use of multiple tables in a pipeline, rather than a single flow table, provides the SDN controller with considerable flexibility
- The OpenFlow specification defines two stages of processing:
  - Ingress processing
  - Egress processing

OpenFlow Protocol

- The OpenFlow protocol describes message exchanges that take place between an OpenFlow controller and an OpenFlow switch
- Typically, the protocol is implemented on top of TLS, providing a secure OpenFlow channel
- The OpenFlow protocol enables the controller to perform add, update, and delete actions to the flow entries in the flow tables
- It supports three types of messages:

Message	Description
Controller-to-Switch	
Features	Request the capabilities of a switch. Switch responds with a features reply that specifies its capabilities.
Configuration	Set and query configuration parameters. Switch responds with parameter settings
Modify-State	Add, delete, and modify flow/group entries and set switch port properties.
Read-State	Collect information from switch, such as current configuration, statistics, and capabilities.
Packet-out	Direct packet to a specified port on the switch.
Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.
Role-Request	Set or query role of the OpenFlow channel. Useful when switch connects to multiple controllers.
Asynchronous-Configuration	Set filter on asynchronous messages or query that filter. Useful when switch connects to multiple controllers.
Asynchronous	
Packet-in	Transfer packet to controller.
Flow-Removed	Inform the controller about the removal of a flow entry from a flow table.
Port-Status	Inform the controller of a change on a port.
Role-Status	Inform controller of a change of its role for this switch from master controller to slave controller.
Controller-Status	Inform the controller when the status of an OpenFlow channel changes. This can assist failover processing if controllers lose the ability to communicate among themselves.
Flow-monitor	Inform the controller of a change in a flow table. Allows a controller to monitor in real time the changes to any subsets of the flow table done by other controllers
Symmetric	
Hello	Exchanged between the switch and controller upon connection startup.
Echo	Echo request/reply messages can be sent from either the switch or the controller, and must return an echo reply.
Error	Used by the switch or the controller to notify problems to the other side of the connection.
Experimenter	For additional functionality.

Table 4.2

OpenFlow Messages

End of Chapter 4

Foundations of Modern Networking

SDN, NFV, QoE, IoT, and Cloud

By: William Stallings

Chapter 5

SDN Control Plane

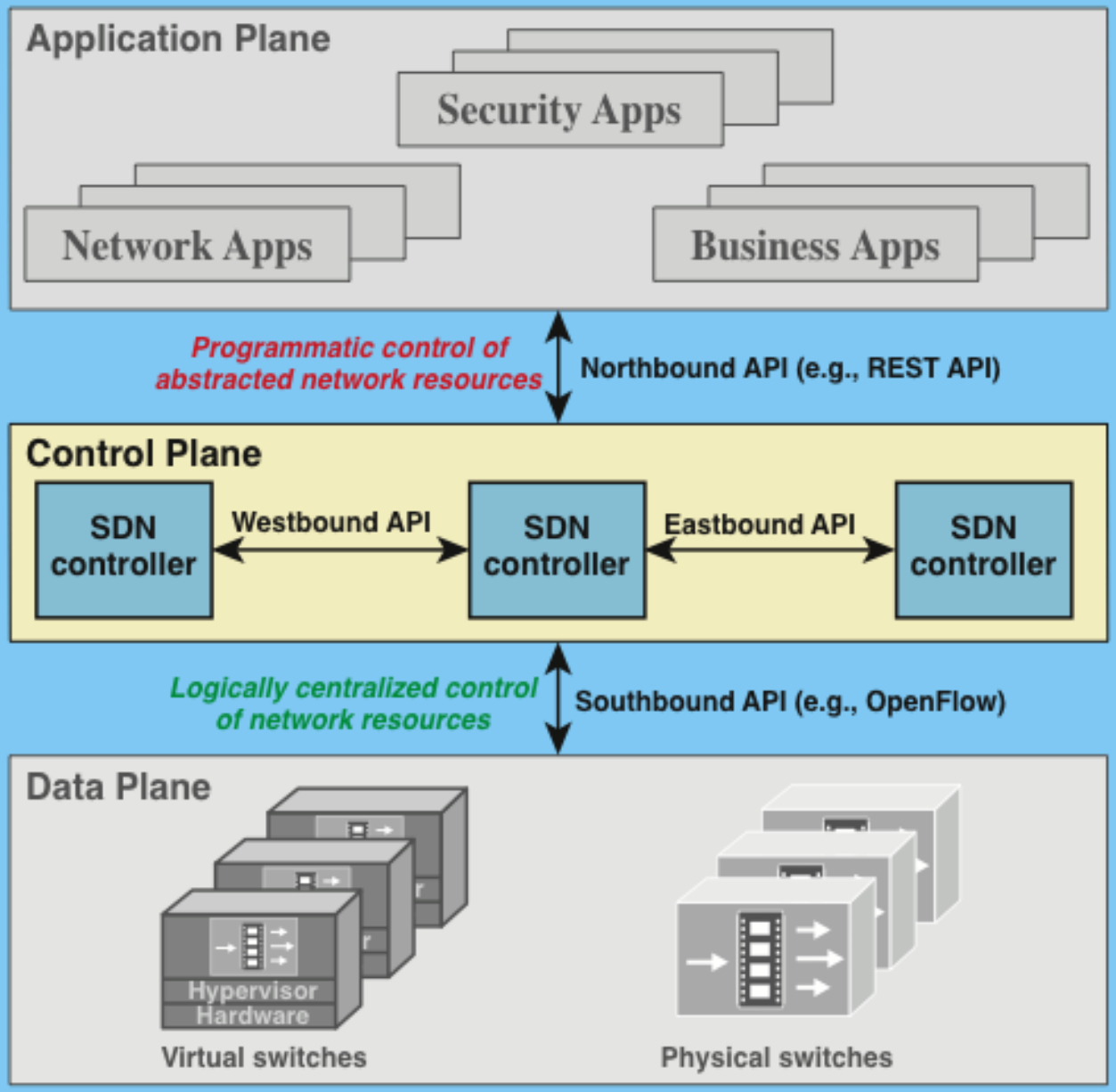


Figure 5.1 SDN Architecture

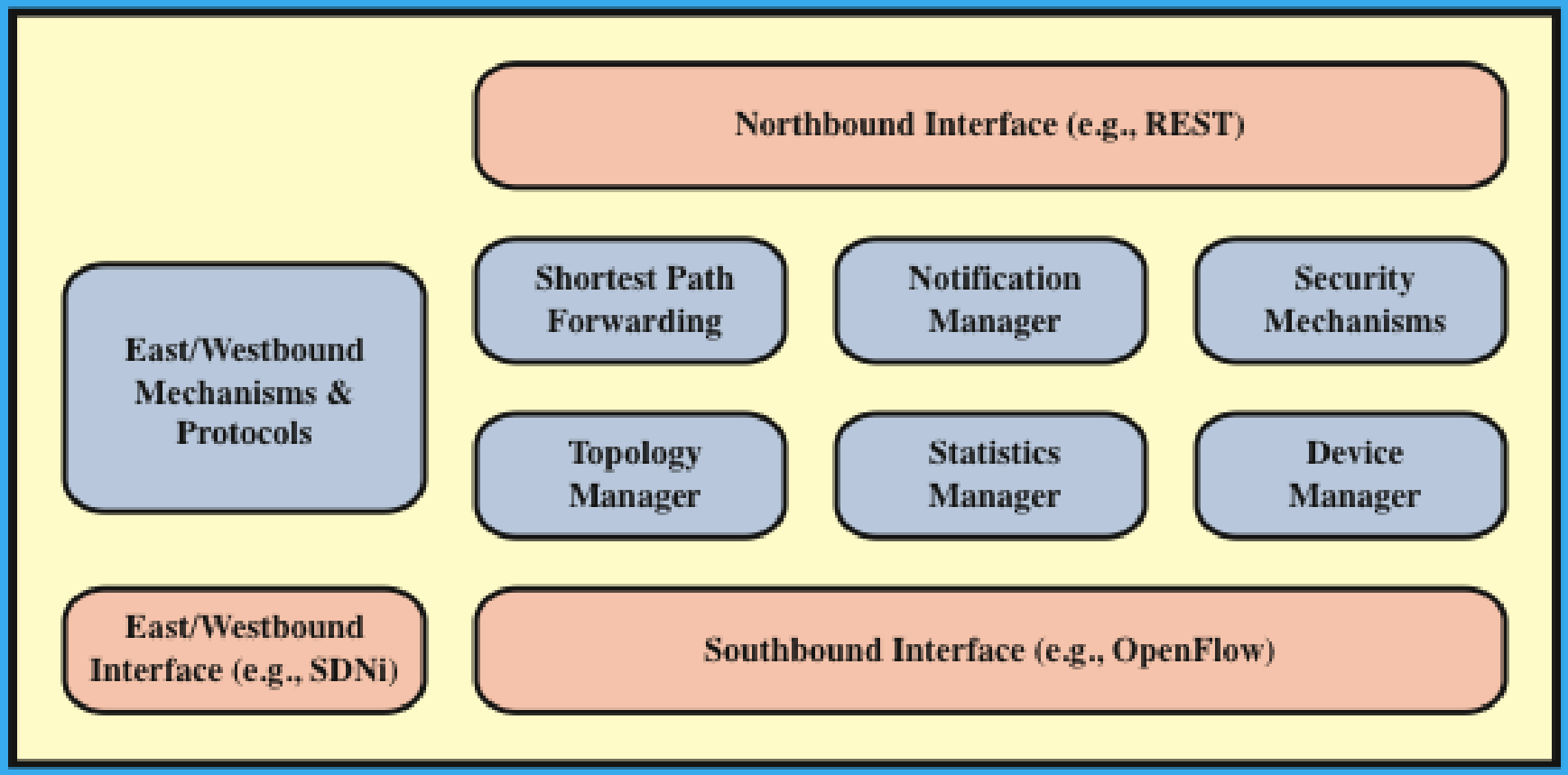
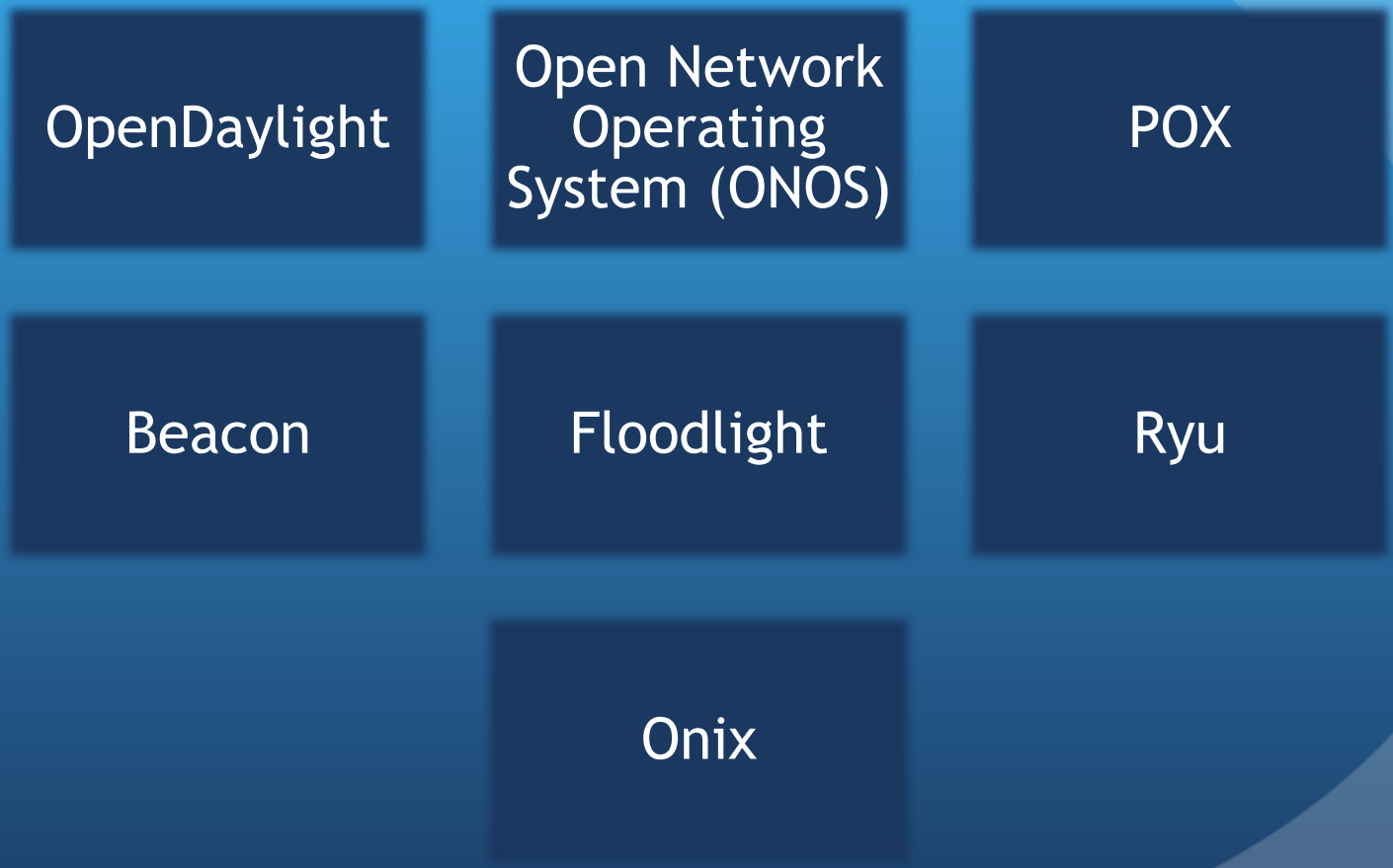


Figure 5.2 SDN Control Plane Functions and Interfaces

Network Operating System (NOS)

- The functionality provided by the SDN controller can be viewed as a network operating system
- A NOS provides essential services, common application programming interfaces (APIs), and an abstraction of lower-layer elements to developers
- The functions of an SDN NOS enable developers to define network policies and manage networks without concern for the details of the network device characteristics

- A number of different initiatives, both commercial and open source, have resulted in SDN controller implementations:



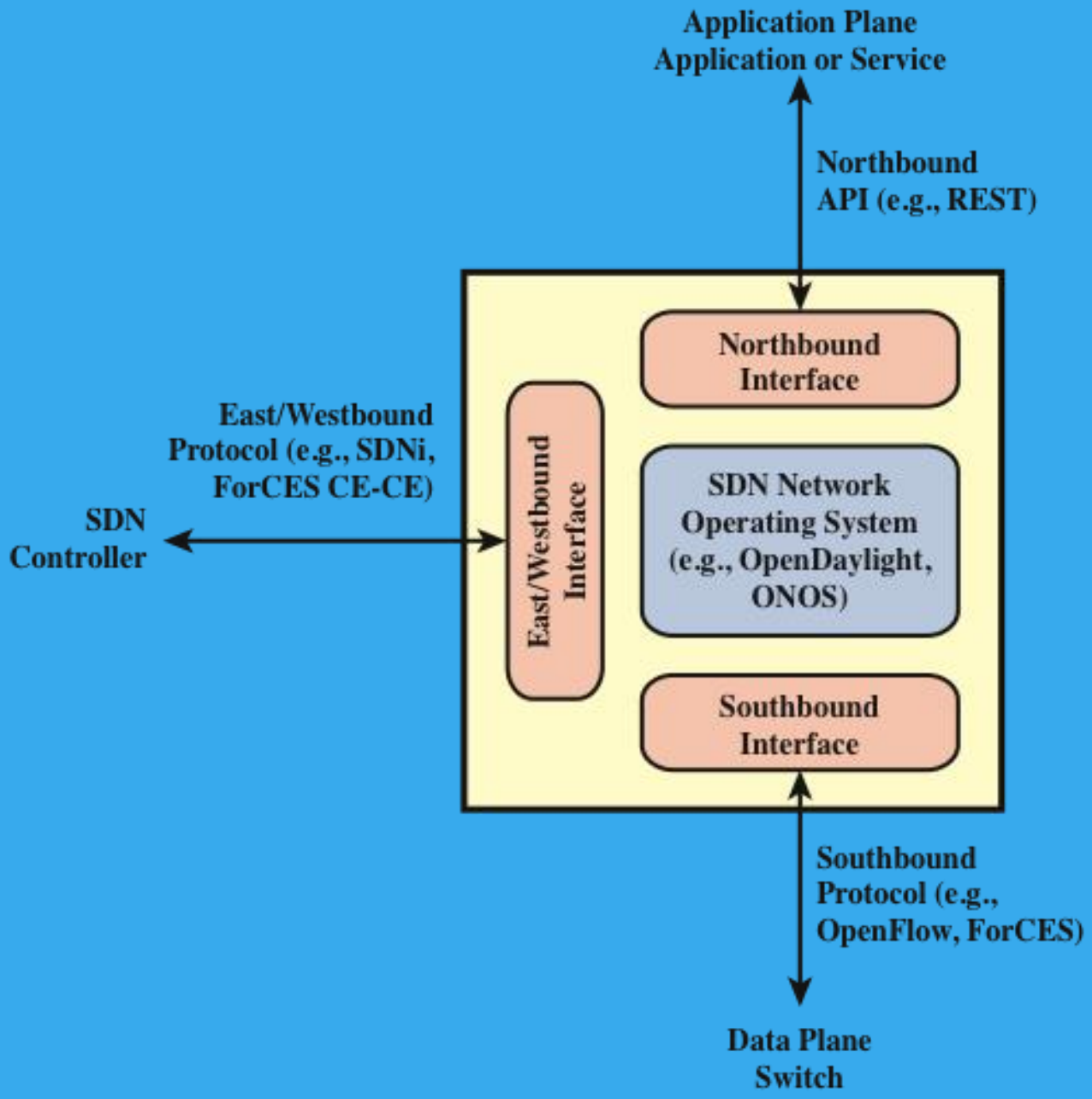


Figure 5.3 SDN Controller Interfaces

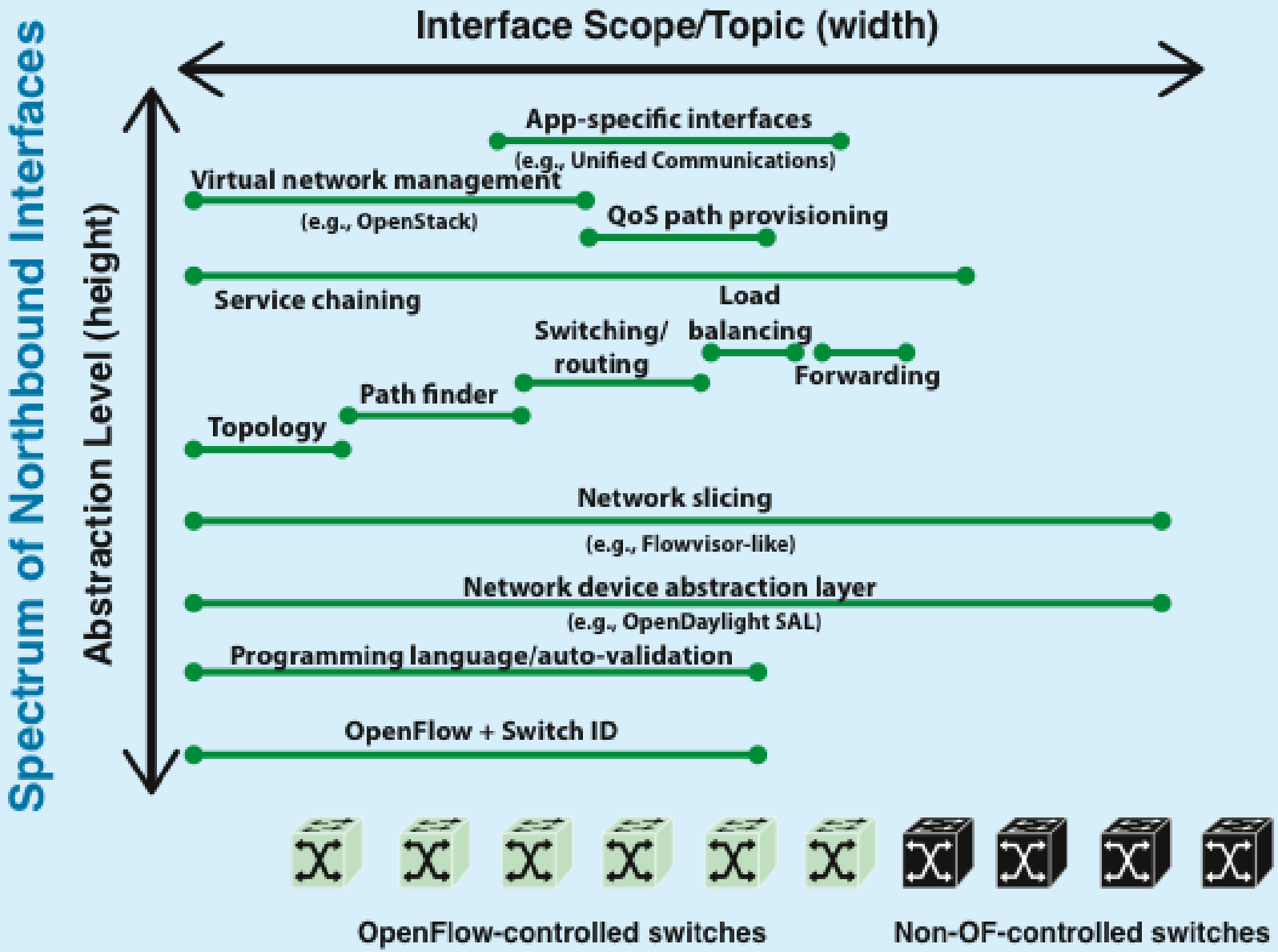


Figure 5.4 Latitude of Northbound Interfaces

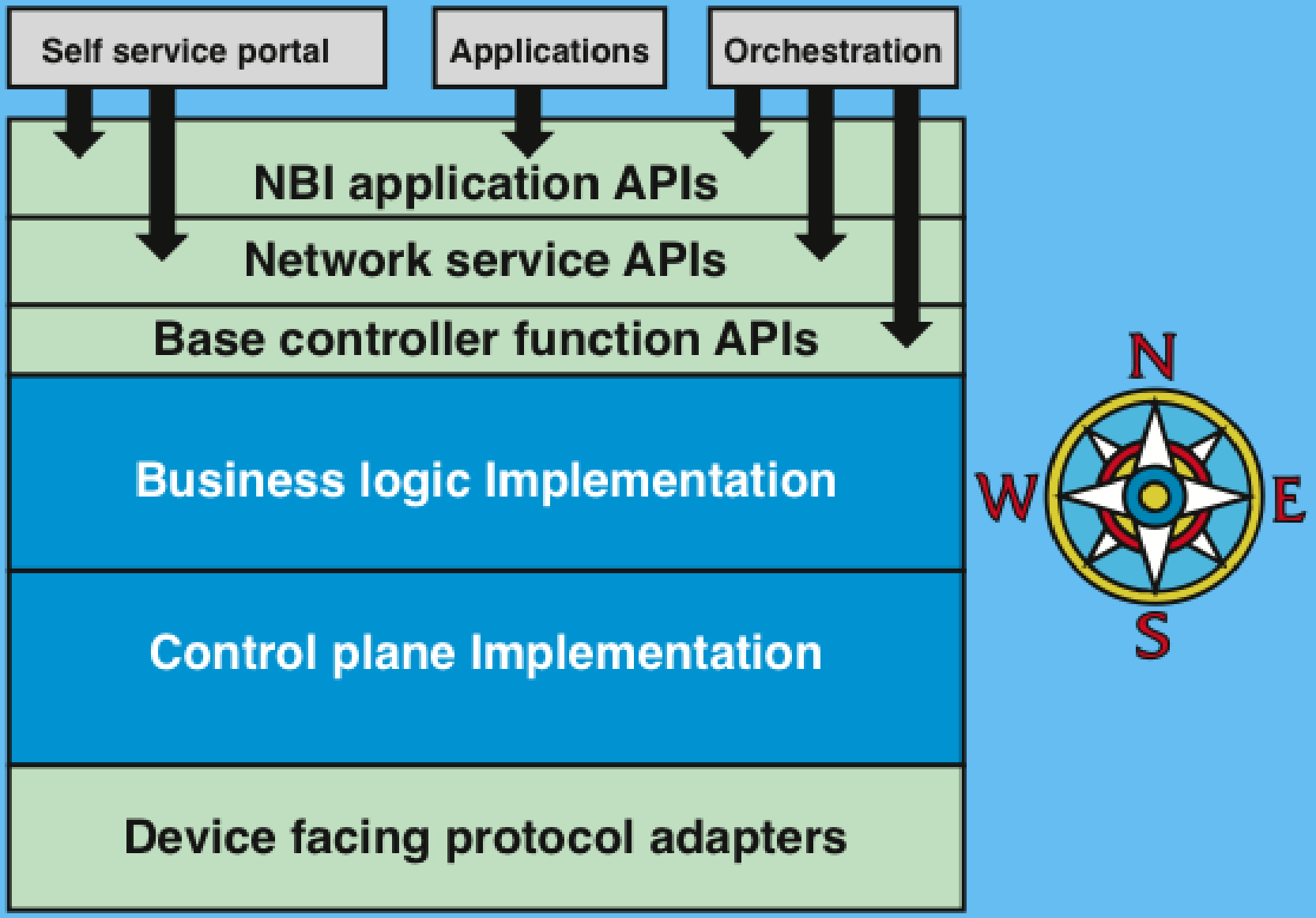
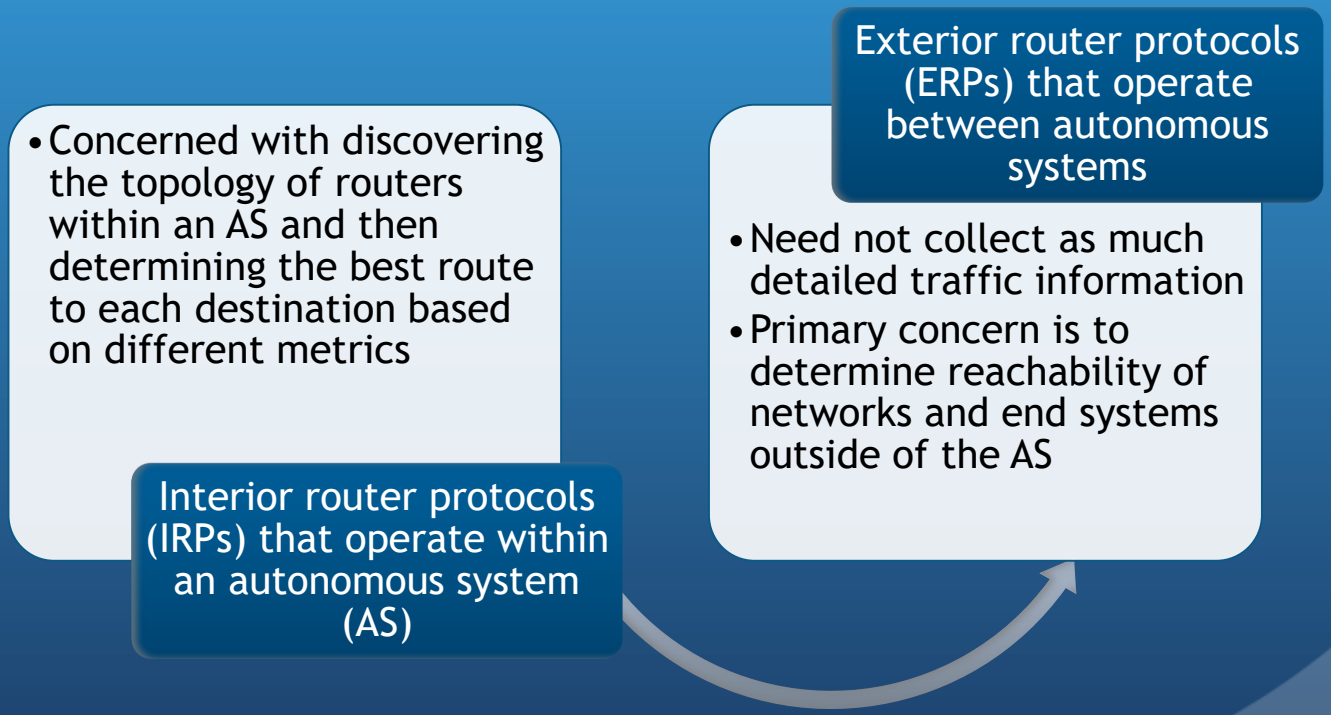


Figure 5.5 SDN Controller APIs

Routing

- The routing function comprises a protocol for collecting information about the topology and traffic conditions of the network, and an algorithm for designing routes through the network
- There are two categories of routing protocols:



Routing

- Traditionally the routing function is distributed among the routers in a network
- In an SDN controlled network, it makes sense to centralize the routing function within the SDN controller
- The controller can develop a consistent view of the network state for calculating shortest paths and can implement application aware routing policies
- The data plane switches are relieved of the processing and storage burden associated with routing, leading to improved performance

Routing

- The centralized routing application performs two distinct functions:
  - Link discovery
    - The routing function needs to be aware of links between data plane switches
    - Must be performed between a router and a host system and between a router in the domain of this controller and a router in a neighboring domain
    - Discovery is triggered by unknown traffic entering the controller's network domain either from an attached host or from a neighboring router
  - Topology manager
    - Maintains the topology information for the network and calculates routes in the network
    - Route calculation involves determining the shortest path between two data plane nodes or between a data plane node and a host

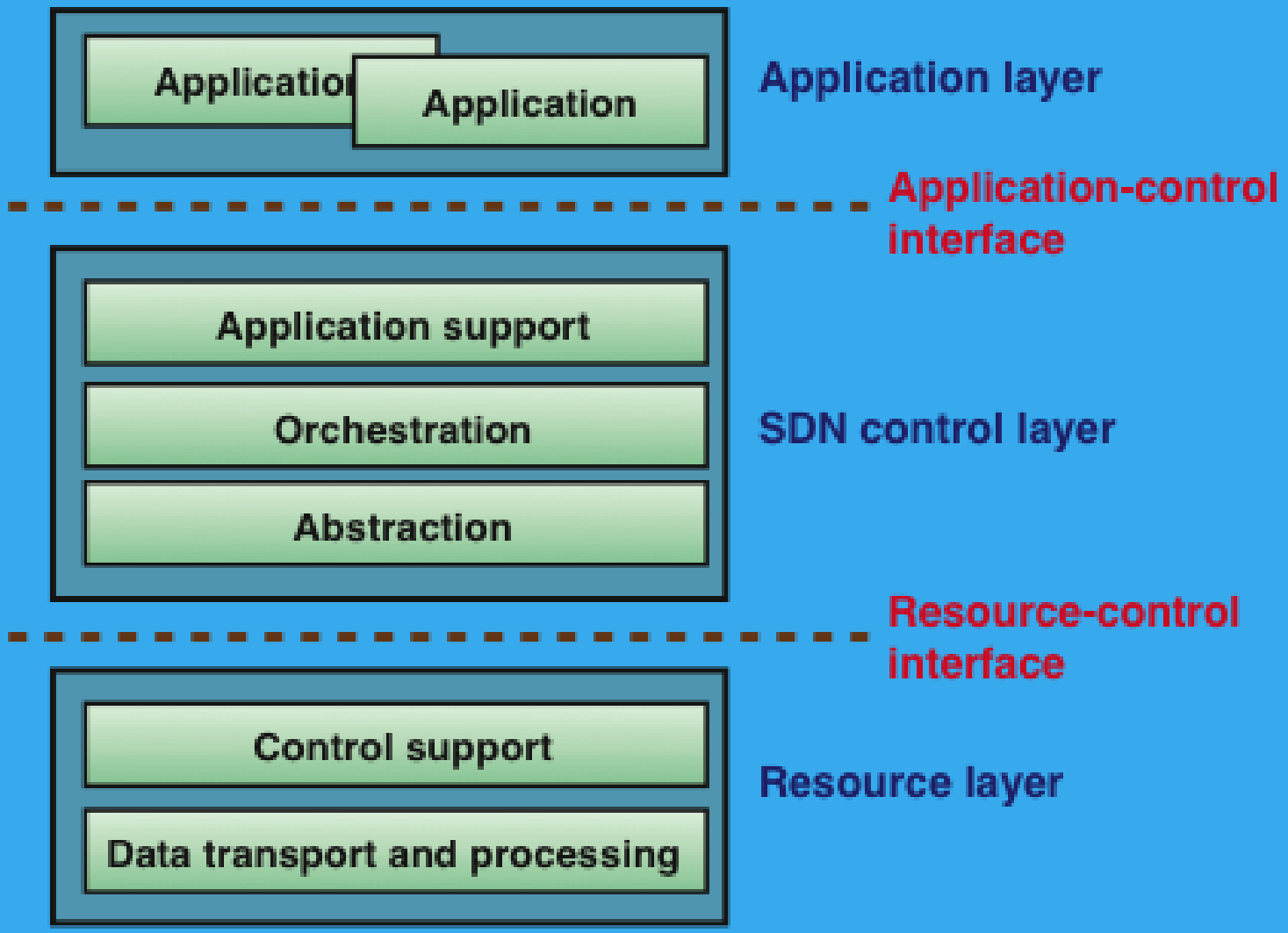


Figure 5.6 High-Level Architecture of SDN (ITU-T Y.3300)

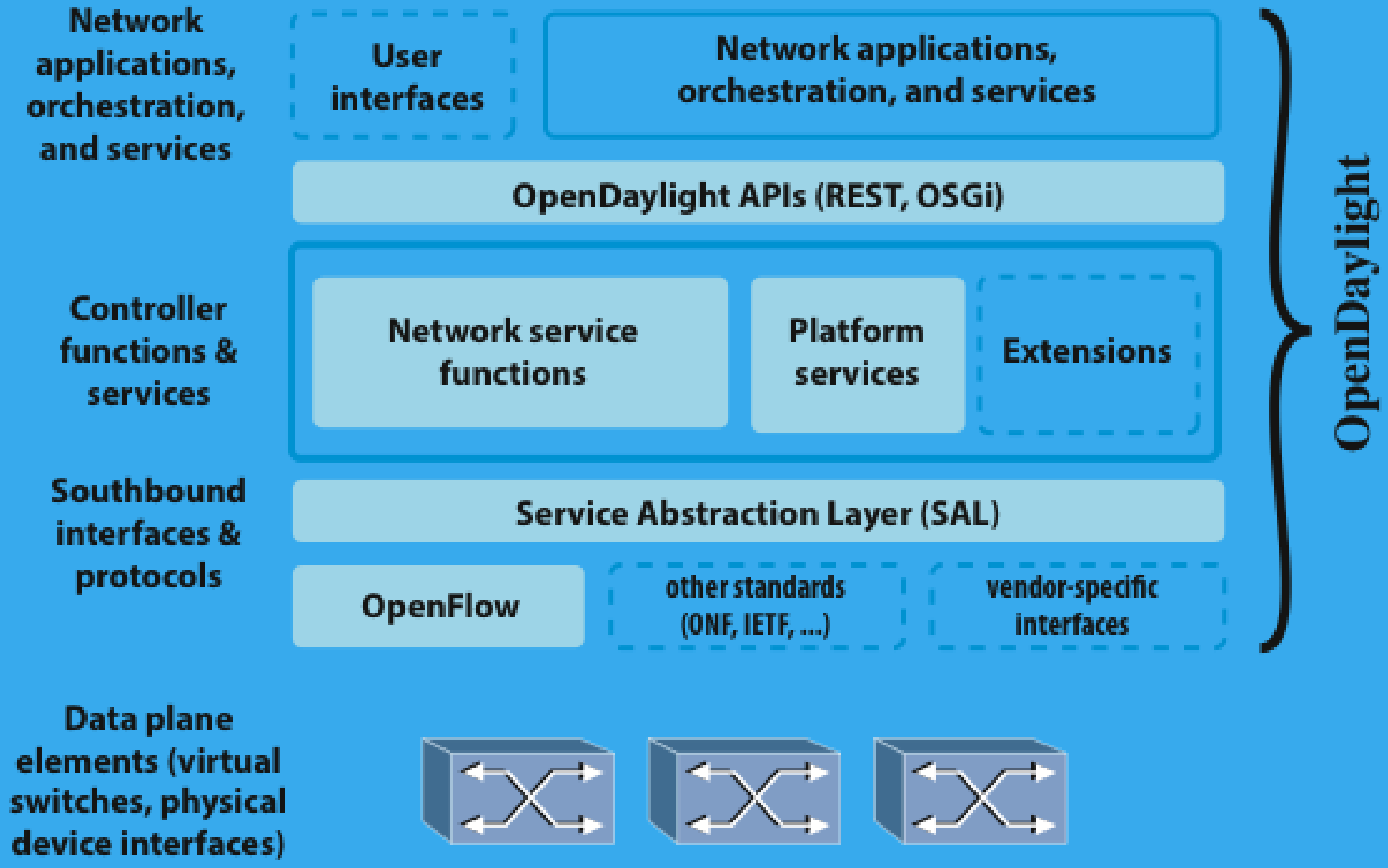


Figure 5.7 OpenDaylight Architecture

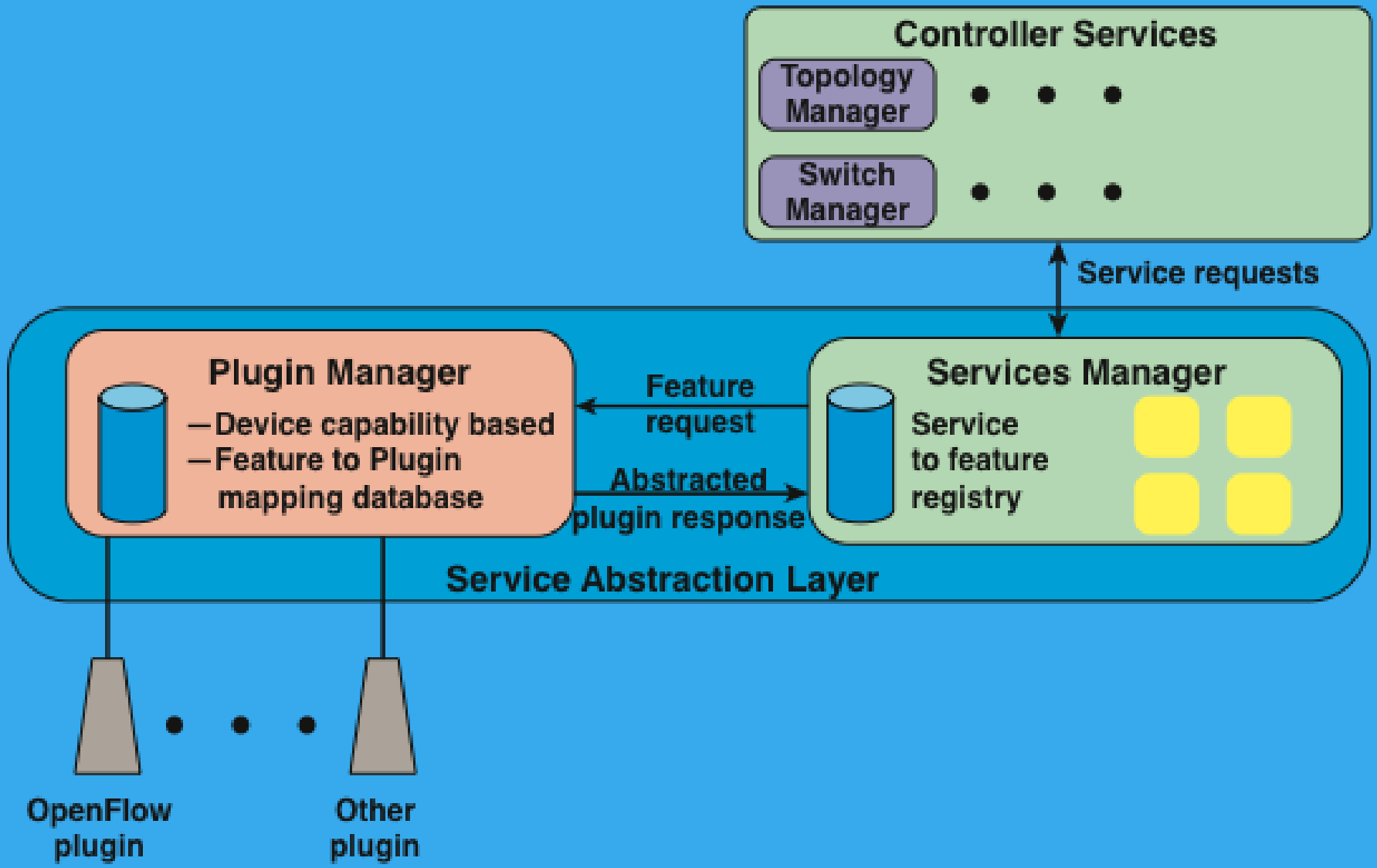


Figure 5.8 Service Abstraction Layer Model

REpresentational State Transfer (REST)

- An architectural style used to define APIs
- This has become a standard way of constructing northbound APIs for SDN controllers
- A REST API, or an API that is RESTful is not a protocol, language, or established standard
- It is essentially six constraints that an API must follow to be RESTful
  - The objective of these constraints is to maximize the scalability and independence/interoperability of software interactions, and to provide for a simple means of constructing APIs

REST Constraints

- The six REST constraints are:

1. Client-server
2. Stateless
3. Cache
4. Uniform interface
5. Layered system
6. Code on demand

Client Server

- This simple constraint dictates that interaction between application and server is in the client-server request/response style
- The principle defined for this constraint is the separation of user interface concerns from data storage concerns
- This separation allows client and server components to evolve independently and supports the portability of server-side functions to multiple platforms

2

•Stateless Constraint

- Dictates that each request from a client to a server must contain all the information necessary to understand the request and cannot take advantage of any stored context on the server
- Similarly, each response from the server must contain all the desired information for that request
- One consequence is that any memory of a transaction is maintained in a session state kept entirely on the client
- Another consequence is that if the client and server reside on different machines, and therefore communicate via a protocol, that protocol need not be connection oriented
- REST typically runs over Hypertext Transfer Protocol (HTTP), which is a stateless protocol